Effects of Self-Explanation in an Open-ended Domain

Amali WEERASINGHE and Antonija MITROVIC
Intelligent Computer Tutoring Group
Department of Computer Science, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
{acw51, tanja}@cosc.canterbury.ac.nz

Abstract: Self-explanation is used in several intelligent tutoring systems in the domains of Mathematics and Physics to facilitate deep learning. Since these domains are well structured, instructional material to self-explain can be clearly defined. We are interested in investigating whether self-explanation can be used in an open-ended domain. For this purpose, we enhanced KERMIT, an intelligent tutoring system that teaches conceptual database design. The resulting system, KERMIT-SE, supports self-explanation by engaging students in tutorial dialogues when their solutions are erroneous. An evaluation study was conducted in July 2002, to investigate whether students will learn better when self-explaining. The results indicate that self-explanation leads to improved performance in both conceptual and procedural knowledge.

1. Introduction

Many Intelligent Tutoring Systems (ITS) have shown significant learning gains for students particularly in the domain of Mathematics [8], Physics [9] and Computer Science [10,14]. However, some empirical studies indicate that students acquire shallow knowledge even in the most effective systems [1]. As the result, students have difficulties in transferring knowledge to novel situations, even though they obtain passing grades on tests. Researchers are therefore interested in finding methods that overcome the shallow learning problem. Self-explanation is described as an "*activity of explaining to oneself in an attempt to make sense of new information, either presented in a text or in some other medium*" [4], and has been shown to facilitate the acquisition of deep knowledge [5]. Since explaining instructional material to oneself facilitates the integration of new information into existing knowledge, self-explanation can be viewed as a knowledge construction activity [4]. However, the results of Chi's study [4] indicated that self-explanation is not merely a process of generating inferences to fill gaps in knowledge, but a process of repairing one's own mental model of the domain. In this context, self-explanation facilitates the identification and removal of misconceptions. Therefore, self-explanation also promotes reflection, which is a meta-cognitive skill many students lack [13].

KERMIT (Knowledge-based Entity Relationship Modelling Intelligent Tutor) [14] is a problem-solving environment that supports students learning database (DB) modelling. In this paper, we describe how we enhanced KERMIT to support self-explanation. Section 2 describes related work. KERMIT is briefly introduced in Section 3 and KERMIT-SE, its enhancement

that facilitates self-explanation, is presented in the next section. The results of the evaluation study are presented in Section 5. The conclusions and directions for future research are given in the final section.

2. Related Work

Self-explanation is facilitated in only a few systems. SE-Coach supports self-explanation by prompting students to explain solved examples [7]. It is implemented within ANDES [9], a tutoring system that teaches Newtonian Physics. The first level of scaffolding in the SE-Coach's interface is provided by a masking mechanism that covers different parts of the example with grey boxes, each corresponding to a unit of information. When the student moves the mouse over a box, it disappears, revealing the content underneath. The second level of scaffolding produces specific prompts to self-explain. Students are prompted to self-explain only when the tutor decides it is beneficial. To determine when to intervene, SE-Coach relies on a probabilistic student model, that monitors how well the student understands the domain by capturing both implicit self-explanations and self-explanations generated through the interface [6]. The results of the empirical evaluation of SE-Coach reveals that the structured scaffolding of self-explanation can be more beneficial in the early learning stages. ANDES has been further enhanced by incorporating ATLAS [15], a module to conduct self-explanation in a natural language. Knowledge construction dialogues (KCDs) which facilitate knowledge construction are currently limited to teaching domain principles. A limited study (with ten participants) revealed that the students interacting with ATLAS learnt significantly more than students who interacted with ANDES [11].

Aleven and Koedigner [1] investigated self-explanation in the PACT Geometry Tutor. The experimental group students provided explanations for solution steps by selecting definitions and theorems from a glossary. The study revealed that explaining reasoning steps results in improved problem solving skills. Students who explained solution steps attempted significantly fewer problems than their peers who provided only the answers, although both groups spent the same amount of time with the tutor. However, there was evidence that self-explainers required fewer problems to reach the tutor's mastery level criterion. PACT Geometry Tutor has been further enhanced to facilitate self-explanation through natural dialogue [2]. The system is currently being developed and an evaluation is yet to be conducted.

These systems use different approaches to facilitate self-explanation, depending on the domain and the target student group. Problem solving activities in these domains are well structured, and the types of self-explanations expected from students can be clearly defined. However, it is challenging to incorporate self-explanation in the domain of database design, as it is an open-ended task: there is an outcome defined in abstract terms, but there is no procedure to find that outcome. It is not sufficient to ask the students to explain the concepts of database modelling, as the database design skills can only be developed through extensive practice.

3. KERMIT: A Knowledge-based ER Modelling Tutor

KERMIT is an ITS aimed at the university-level students learning conceptual database design. The architecture of the system is illustrated in Figure 1. For a detailed discussion of the system, see [14]; here we present some of its basic features. KERMIT is a problem-solving environment in which students practice database design using the Entity Relationship (ER) data model. The system is intended to complement traditional instruction, and assumes that students are familiar with the ER model. The system consists of an interface, a pedagogical module, which determines the timing and content of pedagogical actions, and a student modeller, which analyses student answers and generates student models.

KERMIT contains a set of problems and the ideal solutions to them, but has no problem solver. In order to check the correctness of the student's solution, KERMIT compares it to the correct solution, using domain knowledge represented in the form of more than 90 constraints. It uses Constraint-Based Modelling [13] to model the domain and student's knowledge. Students have several ways of selecting problems in KERMIT. They may work their way through a series of problems, arranged according to their complexity. The other option is a system-selected problem, when the pedagogical module selects a problem for the student on the basis of his/her student model. The interface is composed of three windows tiled horizontally. The top window displays the current problem and provides controls for stepping between problems, submitting a solution and selecting feedback level. The middle window is the main working area. In this window the student draws ER diagrams. Figure 2 represents the interface of KERMIT-SE, which is very similar to that of the original of KERMIT. The only difference is that bottom window of KERMIT has only one section in original KERMIT.

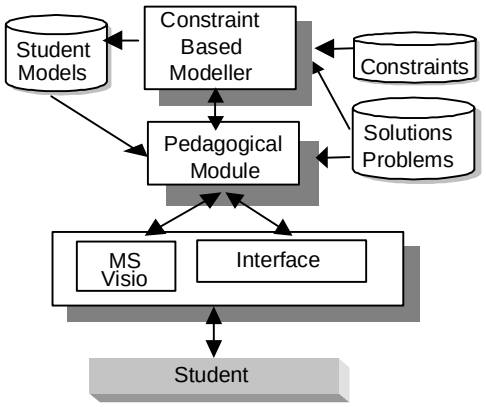

Fig. 1. The architecture of KERMIT

4. Design and Implementation

All systems that facilitate self-explanation prompt students to explain most of the problem-solving steps, requiring students to point out the definitions/theorems used. We believe this approach puts too much burden on able students. Therefore, our tutor prompts for self-explanation only when the student violates a constraint, which indicates missing/erroneous knowledge, or a slip. The tutor is thus able to customise self-explanation based on the student model so that the knowledge construction is facilitated for students who have misconceptions or gaps in their knowledge without disrupting others [3].

For a detailed discussion on how KERMIT was enhanced to support self-explanation-explanation see [16]. Some of important details are discussed here. Since a student can make several errors at each submission, the pedagogical module (PM) needs to decide on which error to initiate the self-explanation process. We have analysed different students' errors and arranged them into a hierarchy. Nodes in this hierarchy are ordered from basic domain principles to more complicated ones. Violated constraints for each type of error are represented as leaves of the hierarchy. Self-explanation is facilitated through tutorial dialogues. We designed a tutorial dialogue for each type of error. The types of dialogues used to support self-explanation-explanation can be divided into two categories: (i) dialogues with a single node, (ii) dialogues with several nodes. Dialogues with a single node handle the basic errors associated with simple connections and are limited to a detailed feedback message. An example of such a dialogue is "*You have connected an entity A to entity B directly. Entities cannot be directly connected to each other.*" Other dialogues are aimed to assist students to understand complex domain concepts. One of the longest dialogues consists of seven levels and 17 nodes.

When the student submits a solution, the student modeller evaluates it against the constraint base and identifies violated constraints. The pedagogical module then searches for the first tutorial dialogue for the same violated constraints. Since the dialogues are ordered according to the complexity of the domain concepts, PM selects the dialogue by traversing the hierarchy in a top-to-bottom, left-to-right manner, selecting the first dialogue that involves some or all violated constraints. In addition, PM produces an overall view of the solution by developing a list of general feedback messages for each violated constraint, displayed in the

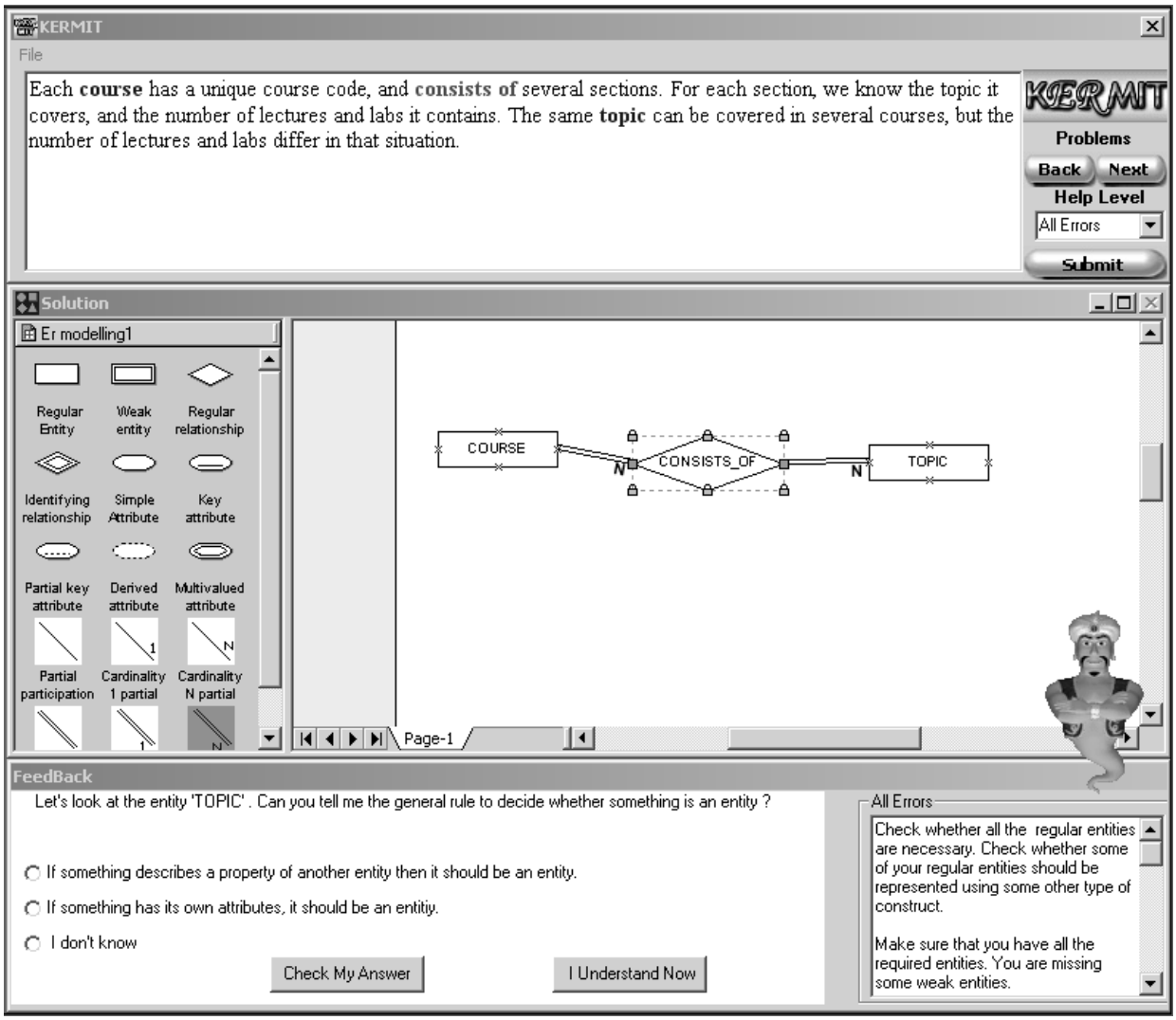

Fig. 2. Interface of KERMIT_SE

right hand side of the bottom window (Figure 2). On the left hand side of the feedback window, the error associated with the chosen dialogue is displayed. Students can initiate self-explanation using the *More Help* button. This way students are given more control on deciding when to go through self-explanation-explanation dialogues.

For example, consider the problem statement shown in Figure 2. The student's solution contains several errors: the *TOPIC* entity should be represented as a partial key of the SECTION weak entity (which is missing), COURSE is missing the key attribute, the CONSISTS_OF relationship should be an identifying relationship and its cardinality is wrong. The ideal solution for this problem is shown in Figure 3. The pedagogical module selects the dialogue corresponding to the simplest error (TOPIC represented as an entity instead of as an attribute) to start with. Figure 4 presents a sample dialogue, which may occur between the student and the tutor. Initially, the student is informed of the error, and is asked to interpret the action in the context of ER modelling (*tutor-1*). A list of possible answers is provided from which the correct one can be selected. If the student fails to provide the correct answer or indicates that s/he needs more help (*student-1*), s/he will be asked a more specific question that provide a further opportunity to understand the fundamental principle that is violated (*tutor-2*). However, if s/he fails to identify the mistake and correct it, even after going through a series of detailed questions, as the last resort the tutor will specify the correct action to be performed together with a brief description about the fundamental principle that needs to be learnt (*tutor-7, tutor-8*). KERMIT-SE uses various types of interactions such as simple questions (*tutor-1*), fill-in-a-blank (*tutor-8*), or true-false questions, to motivate the student to self-explain. Since all dialogues initiate self-

explanation by pointing out the error (*tutor-1*), students are given an opportunity to reflect on their problem solving procedure, which is another important meta-cognitive skill.

The short-term student model in KERMIT consists of a list of satisfied and a list of violated constraints for the student's solution, while the long-term model records the history of each constraint (how often a constraint was relevant, and how often it was satisfied/violated) [15]. In KERMIT-SE, the long-term model additionally records the types of errors made and the level of prompting the student needed to correct his/her mistake for every constraint.

Tutor-1: You've modelled *TOPIC* as an entity. What does that mean in terms of properties of an entity
Student-1*: I don't know*
Tutor-2: Let's try a simple thing. How do we model characteristics of an entity?
Student-2*: Using Attributes*
Tutor-3: Great! Now tell me why modeling *Topic* as an entity contradicts with the problem description?
Student-3*: I'm not sure*
Tutor-4: Don't worry. I'm here to help you. Check carefully whether you can find any attributes for *Topic*. {After sometime} Please tell me whether you managed to find any attributes
Student-4*: No*
Tutor-5: Fine. It doesn't have any attributes. So now can you tell me why modeling *Topic* as an entity contradicts with the problem description?
Student -5*: Because it doesn't have any attributes to describe its characteristics*
Tutor-6: Great! Now tell me the best way to model *Topic*?
Student-6*: I need more help*
Tutor-7: Ok. If *Topic* has no attributes, it cannot be an entity. So it should either be a relationship or an attribute. So what is the best way to model *Topic?*
Student-7*: Attribute*
Tutor-8: Great! Now use the correct word to fill in the blanks so that you'll remember what you learnt. …….. are used to describe characteristics of an …….. in a ER model
Student-8*: attributes, entity*
Tutor-9: Excellent, Go ahead and do the changes. Submit the solution when you're done.

Fig. 4. A sample tutorial dialogue

5. Evaluation Study

An evaluation study was conducted in July 2002 with students enrolled in an introductory database course at the University of Canterbury. We wanted to test the hypothesis that self-explanation-explanation facilitates acquisition of both procedural and conceptual knowledge. The experimental group used KERMIT-SE, while the control group used a cut down version of KERMIT. Both groups received the list of all errors for their solutions, and could ask for the ideal solution. Even though there were five different levels of feedback in the original KERMIT, only *Detailed Hint* was available to the control group to make it comparable with the experimental group. As the name suggests, *Detailed Hint* provides a details feedback message on a single error in a student solution.

The experiment was carried out during normal lab hours over the duration of two weeks, and consisted of four phases: pre-testing, system interaction, post-testing and subjective system assessment. The pre- and post-tests consisted of two questions each, of equal difficulty. The first question required students to design an ER model for the given requirements, whereas the second question required them to explain the design decisions for the given ER model. The tests were rotated between successive sessions to minimise any effect resulting from variation in test difficulty. Ideally, each student was expected to spend a total of 4 hours to complete the four stages of the study.

We developed two different versions of the questionnaire: control group was given 12, and the experimental group 15 questions. Initially, students were asked about their previous experience in database modelling, and their impressions of the system and its

interface. The experimental group was additionally asked about their perception of self-explanation support. Some of the questions asked were how easy/difficult the dialogues were to understand, and how useful the dialogues were for understanding errors. Students answered on a Likert scale with five responses ranging from *very poor* (1) to *very good* (5), and were also allowed to give free-form responses.

5.1. Objective analysis

Table 1 gives a few statistics about the study. The sizes of control and experimental group differ, as they depended on how many students turned out to corresponding lab sessions. The mean score on the pre-test for all students was 72.18% (SD=18.72%). The difference in pre-test scores for the two groups is insignificant, confirming that the groups are comparable. Examining the logs of the session, we see that only 19 students in the experimental group have gone through at least one dialogue, (they had control over that via the *More Help* button). We are interested in these students, as the rest of the experimental group has not self-explained, and we summarize statistics for those students separately (*self-explainers*). The mean score on the pre-test for self-explainers is significantly higher than the mean for the control group. Therefore, we cannot directly compare the control group to self-explainers. However, the other students in the experimental group, who have not gone through any of the dialogues (column *non self-explainers* in Table 1), are comparable to the self-explainers, as there is no significant difference for these two groups of students on the pre-test.

There is no significant difference between the problem solving time for the control and the experimental groups. However, self-explainers spent significantly more time on problem solving ($t=5.01$, $p<0.001$) than non self-eplainers. This might be due to the self-explanation dialogues, as student needed time to answer the questions. However, the self-explainers also attempted and solved significantly more problems than the rest of the experimental group. Therefore, self-explanation supports problem-solving.

	Control	**Experimental**	**Self-explainers**	**Non self-explainers**
No of students	72	53	19	34
Problem solving time (min.)	105:21 (44:19)	98:37 (49:34)	133:21(30:44)	79:13 (47:41)
No. of attempted problems	7.08 (2.69)	6.34 (3.22)	8.21 (2.42)	5.29 (3.17)
No. of completed problems	5.25 (2.43)	4.62(2.63)	6.36 (2.31)	3.65 (2.29)
No. of post-tests	59	35	18	17
Pre-test	70.98 (18.47)	75.61 (17.33)	79.32 (13.16)	73.17 (20.47)
Post-test	79.94 (17.75)	78.11 (14.35)	79.76 (12.22)	77.37(16.76)

Table 1. Mean system interaction details

The participation in the experiment was voluntary, and not all students completed the study. We report the number of post-tests in Table 1. The difference between the post-test scores of the experimental and control groups is not significant. The control group students improved significantly on the post-test ($p=<0.01$). However, these students had lowest existing knowledge (lowest pre-test score) and therefore had more room for improvement. Even though the self-explainers did better in the post-test, the improvement is not significant. As a result, the difference in gain scores for the experimental and the control groups was significant ($t=1.33$, $p=0.09$). The self-explainers have improved only slightly on the post-test, and their performance on the conceptual question increased from 83.70 to 84.81 (not significant).

The self-explainers on average went through 6.95 dialogues, ranging from 1 to 21. On average, students completed 78.25% of the dialogues, with an average of 57.61% of correct responses to the questions in the dialogues. To test the second part of our hypothesis (self-explanation results in improved conceptual knowledge), we analysed the student answers to the first question of a chosen dialogue, which prompts students to explain domain concepts. Figure 5 illustrates the correctness of students' explanations. The probabilities of correct answers on the first and subsequent occasions were averaged over all error types and over all students. The fit to the power curve is very good, indicating that students learn by explaining.

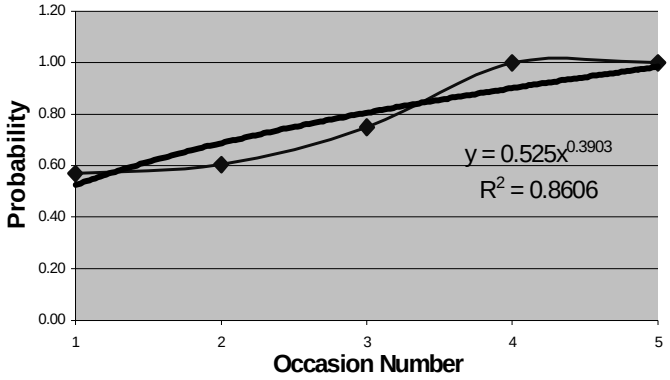

Fig. 5. Performance on the first question in the dialogues

5.2. Questionnaire Analysis

A summary of the responses to the user questionnaires is given in Table 2. Students in both groups required approximately the same time to learn the interface. This was expected, as there was not much difference between the two interfaces. Students in the control group found it significantly easier to use the interface. This might be due to the fact that students in the control group had more features in their version due to the self-explanation support provided. The difference in mean responses on the amount learnt and the enjoyment were not significant.

Control group student found feedback to be useful, even though they only had limited feedback. Although student on average find questions in the dialogue difficult to understand, they rated the usefulness of the dialogues higher than the rank of feedback by the control group. 68% of the control group students indicated that they would recommend KERMIT to other students while the percentage of experimental group students who had the same opinion was lower (60%). 60% of the control group students preferred more details in the feedback whereas, only 17% indicated that they do not want more details. When asked how many questions they had to go through on average to realise a mistake in their student solution, the majority (55%) of experimental group students indicated that 2 to 3 questions were needed. Only 3% of the experimental group students indicated that they needed to go through the entire dialogue. This suggests that being able to resume problem solving in the middle of a dialogue might have increased the usability of KERMIT-SE.

Table 2. Means for the questionnaire responses

	Control group	Experimental group
Time to learn the interface	14.27 (13.99)	11.82 (11.64)
Amount learnt	3.35 (0.82)	3.33 (0.78)
Enjoyment	2.84 (0.98)	2.72 (1.14)
Ease of using interface	2.78 (0.98)	2.34 (0.87)
Usefulness of feedback	3.45 (1.09)	N/A
Ease of understanding the questions in the dialogues	N/A	2.65 (0.95)
Usefulness of dialogues to understand the mistake	N/A	3.50 (0.82)

5. Conclusions and Future Work

Self-explanation is an effective learning strategy to facilitate deep learning. This research focuses on incorporating self-explanation into a tutor that teaches the open-ended task of ER modelling. KERMIT-SE supports self-explanation by engaging students in tutorial dialogues about errors they make. Students are asked problem-specific and general questions, and can select answers from menus.

An evaluation study was conducted in July 2002, to investigate whether guided self-explanation would improve students' learning in the domain of database modelling. The experiment involved second-year university students enrolled in an introductory database course. The results indicate that self-explanation leads to improvement in problem solving and in answering questions about domain knowledge.

We plan to enhance the student model in KERMIT-SE to provide adaptive self-explanation. i.e. to provide support self-explain based on the student's existing self-explanation skill. The enhanced student model can also be used to provide additional support in acquiring domain knowledge to students who have difficulty in understanding domain concepts.

Acknowledgements: We thank Pramuditha Suraweera and Danita Hartley for their help in implementing KERMIT-SE. This research was made possible by the NZODA postgraduate scholarship awarded to the first author.

References

1. Aleven, V., Koedinger, K. R. and Cross, K. Tutoring Answer Explanation Fosters Learning with Understanding. In: Artificial Intelligence in Education, Lajoie, S.P. and Vivet, M.(eds.), Amsterdam : IOS Press (1999) 199-206.
2. Aleven, V., Popescu, O. and Koedinger, K. R. Towards Tutorial Dialogue to Support Self-Explanation: Adding Natural Language Understanding to a Cognitive Tutor. Int. Journal on Artificial Intelligence in Education, 12 (2001) 246-255.
3. Bunt, A. and Conati, C. Modeling Exploratory Behaviour. In: Bauer, M., Gmytrasiewicz, P. J. and Vassileva, J. (eds.), Proc. of 8th International Conference, User Modeling, Sonthofen, Germany (2001) 219-221
4. Chi, M. T. H. Self-explaining Expository Texts: The dual processes of generating inferences and repairing mental models. Advances in Instructional Psychology, (2000) 161-238.
5. Chi, M. T. H., Bassok, M., Lewis, W., Reimann, P. and Glaser, R. Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. Cognitive Science, 13 (1989) 145-182.
6. Conati, C. and VanLehn K., Providing Adaptive Support to the Understanding of Instructional Material. In Proc. IUI 2001 Sante Fe, New Mexico (2001).
7. Conati, C. and VanLehn, K. Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. Int. J. Artificial Intelligence in Education, 11 (2000) 389-415.
8. Corbett, A.T.M., Trask, H.J., Scarpinatto, K.C. and Handley, W.S. A formative evaluation of the PACT Algebra II Tutor : support for simple hierarchical reasoning. Proc. ITS'98 374-383.
9. Gertner A. S. and VanLehn, K. ANDES: A Coached Problem-Solving Environment for Physics, In Gauthier G., Frasson, C. and VanLehn, K. (eds.), Proc. ITS 2000, Montreal (2000) New York : Springer 133-142.
10. Grasser, A. C., Wiemer-Hastings, K. Wiemer-Hastings, P. and Kreuz, R., Tutoring Research Group 1999. AUTOTUTOR: A Simulation of a Human Tutor. Journal of Cognitive Systems Research 1(1) (1999) 35-51.
11. Grasser, A.C., VanLehn, K., Rose, C.P., Jordan, P.W. and Harter, D. Intelligent Tutoring Systems with Conversational Dialogue, AI Magazine, American Association for Artificial Intelligence, Winter 2001, 39 -51
12. Mitrovic, A. Investigating Students' Self-assessment Skills. In Bauer, M., Gmytrasiewicz, P. J. and Vassileva, J. (eds.), Proc. UM 2001, Berlin Heidelberg (2001) Springer-Verlag 247-250.
13. Ohlsson, S. Constraint-based Student Modelling. In: Greer, J.E., McCalla, G (eds) Proc. of Student Modelling: the Key to Individualized Knowledge-based Instruction, Springer-Verlag Berlin (1994) 167-189.
14. Suraweera, P. and Mitrovic, A. KERMIT: a constraint-based tutor for database modelling. In: S. Cerri, G. Gouarderes and F. Paraguacu (eds.) Proc. ITS'2002, Biarritz, France, LCNS 2363, 2002, 377-387.
15. VanLehn, K. et.al Fading and Deepening: The Nest steps for ANDES and Other Model-Tracing Tutors. In Proc. ITS 2000, Gauthier, G., Frasson, C. and VanLehn, K. (eds.), Montreal (2000) 474-483.
16. Weerasinghe, A., Mitrovic, A. (2002) Enhancing learning through self-explanation. Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson, C-H Lee (eds.) Proc. ICCE 2002, 244-248.